# תרגיל מס' 2 Desktop Facebook

בחרנו לממש את התבניות עיצוב הבאות:

1. Transparent Façade

הסבר: מטרת התבנית היא להקל את העבודה מול תת מערכות אשר מטפלות באוספים שונים של מידע שהממשק משתמש בהם.

בחרנו בתבנית זו כדי לפשט עבור הטופס את תהליך הבאת המידע מה API ועבודה מול המחלקות השונות בפרויקט, כתוצאה מכך ישנו ממשק פשוט יותר לשימוש אשר ניתן להחליפו לגמרי אם נדרש.

בחרנו לממש את התבנית בצורה 'שקופה' יחסית על מנת לאפשר את ביצוע כל תהליך הבאת המידע בפעולה אחת או לחילופין חלקים ממנו; לדוגמא להביא רק הודעות משתמש ולא גם את ימי ההולדת.

מימשנו את התבנית ע"י מחלקה אחת,*InfoFetch* ,שמרכזת את פונקציונליות של א) הבאת מידע (כמו הודעות מהתיבת דואר-נכנס) בתצורת מבני הנתונים של ה-API ו-ב) תרגומו למבני נתונים שניתן להשתמש בהם בממשק המשתמש, *DataTable*.   
מבחינת הממשק, כל שעליו לבצע זה להפעיל את המתודה *GetAll* ולאחר סיומה יופיעו במבנה נתונים של *InfoFetch* כל הטבלאות עם המידע של המשתמש. בזכות תבנית זו, הצלחנו להכמיס את הלוגיקה של הבאת מידע מה-API ותרגומו.

המחלקות המשתתפות:

**Sub-system Classes**:

Event,

BirthdayHolder,

MessageFilterer,

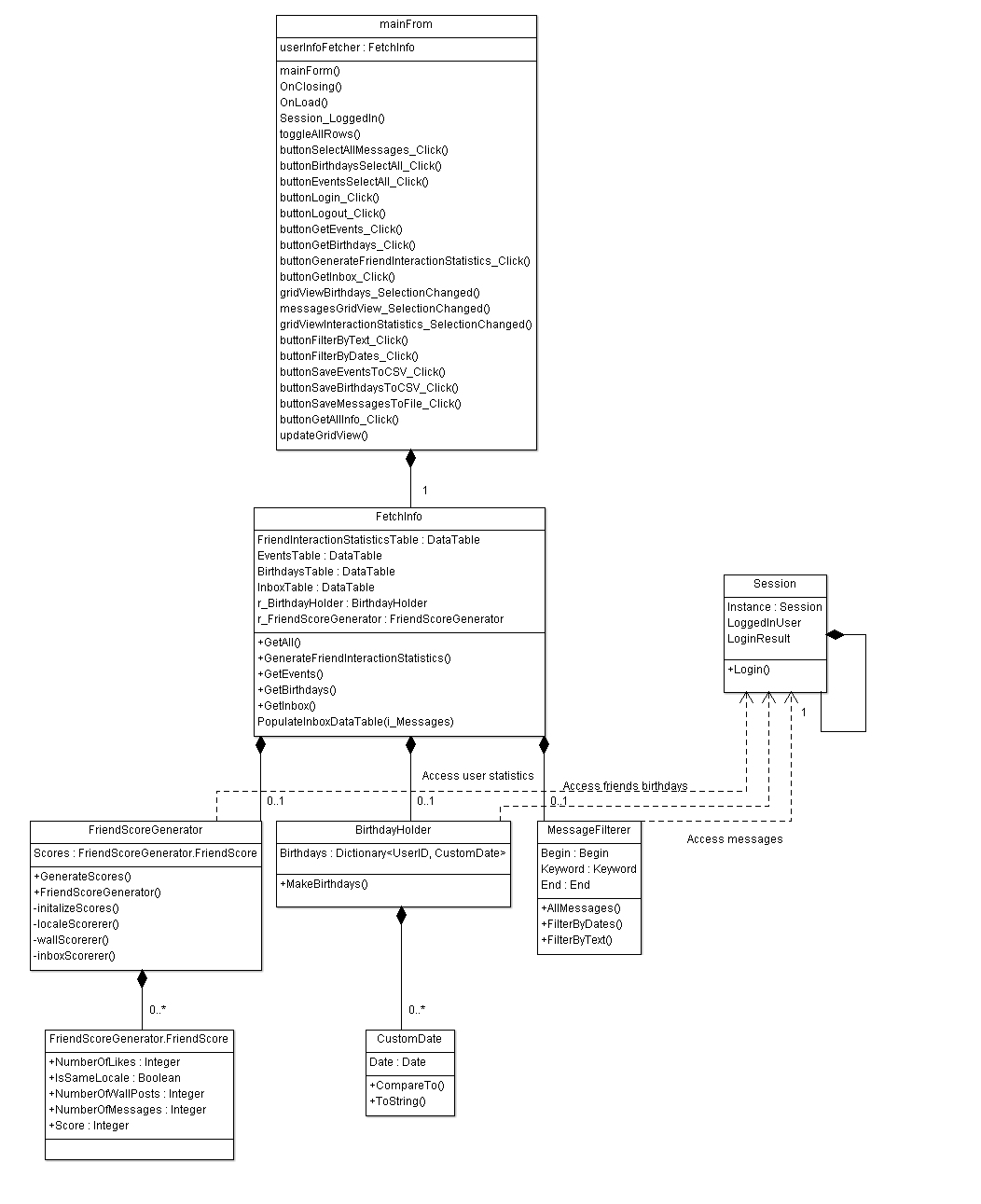
FriendScoreGenerator

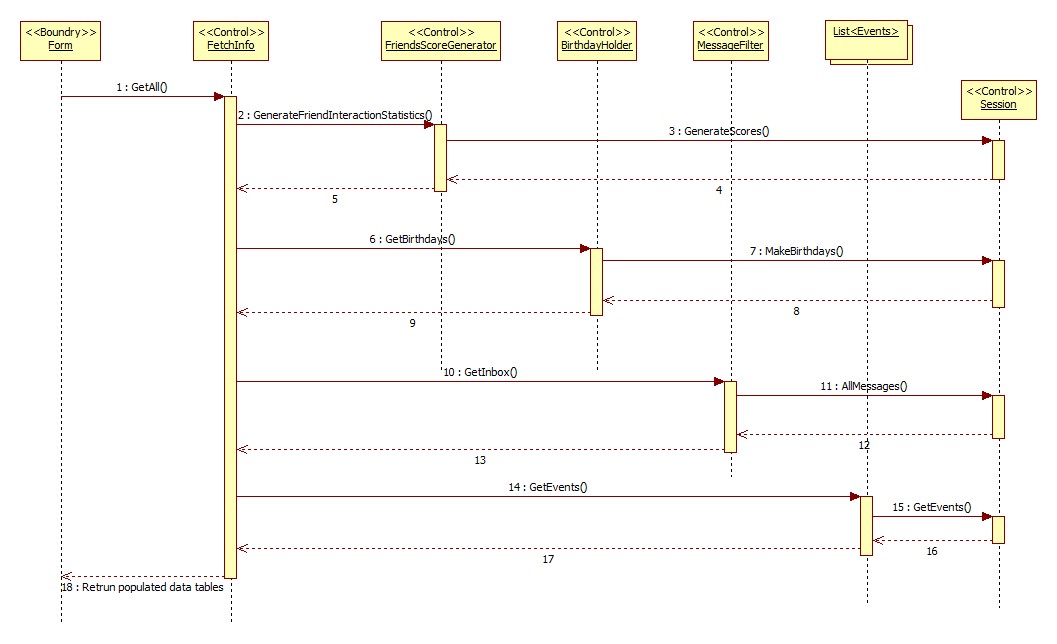
**Façade:**

FetchInfo

**Client:**

MainForm





1. Factory

בפרוייקט שלנו, מנגנון הסריאליזציה לפורמט csv ממומש על ידי מחלקה אבסטרקטית *AbstractCSVSerializer* שממנה יורשות מחלקות שמממשות סריאליזציה ספציפית עבור הודעות, ימי הולדת ואירועים. על מנת לבטל את ההכרות של הממשק עם המחלקות הקונקרטיות, הכמסנו את תהליך יצירת אובייקט סריאליזציה במחלקה *CSVSerializerFactory*.

בצורה כזו יהיה ניתן להוסיף בקלות יחסית עוד סוגים של סריאליזציות (למשל, רשימת כתובות אימייל של חברים) על ידי שינוי קל במחלקת *Factory*.

מחלקות משתתפות:

**Product:**

AbstractCSVSerializer

**Concrete Product:**

CSVBirthdaySerializer,

CSVEventSerializer,

CSVMessagesSerializer

**Creator:**

CSVSerializerFactory

